

Aims

This exercise aims to get you to:

- Install and configure Hadoop MapReduce
- Practice HDFS operations
- Test Hadoop MapReduce with the pseudo-distributed mode

Background

In the examples below, we have used the `$` sign to represent the prompt from the command interpreter (shell). The actual prompt may look quite different on your computer (e.g. it may contain the computer's hostname, or your username, or the current directory name). Whenever the word “edit” is used, this means that you could use your favorite text editor (e.g. vim, emacs, gedit, etc.).

The virtual machine image was created 2 years ago, and thus hadoop-2.7.2 was installed. However, if you want to try other versions (e.g., hadoop 2.9.1), you can download the package and install it by following the lab instructions as well.

The virtual machine image has been pre-configured for you. If you want to install Hadoop on your own computer, you need to do more configurations before following today's lab. Please refer to <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html> for more details.

Start up the virtual machine

Login the lab computer using your CSE account

Start the virtual machine in a terminal using the following command:

```
$ vm COMP9313
```

A virtual machine running Xubuntu 14.04 should be started. Both user name and password is comp9313. The sudo password is also comp9313 in the system.

The virtual machine image has been made persistent due to some security reasons, which means that after you restart your lab computer, anything you did in it will be lost. Today's lab aims to let you know how to install and configure Hadoop. In future labs, Hadoop will be ready for you to use.

Configure Hadoop and HDFS

1. Download Hadoop and Configure HADOOP_HOME

```
$ mkdir ~/workdir
```

Then get into the directory using:

```
$ cd ~/workdir
```

Download the Hadoop package by the command:

```
$ wget https://archive.apache.org/dist/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz
```

Then unpack the package:

```
$ tar xvf hadoop-2.7.2.tar.gz
```

Now you have Hadoop installed under `~/workdir/hadoop-2.7.2`. We need to configure this folder as the working directory of Hadoop, i.e., `HADOOP_HOME`.

Use the following command to install gedit if it is not installed yet (sudo password is comp9313):

```
$ sudo apt-get install gedit
```

Open the file `~/.bashrc` using gedit (or use vim or emacs if you are familiar with them):

```
$ gedit ~/.bashrc
```

Then add the following lines to the **end** of this file:

```
export HADOOP_HOME=/home/comp9313/workdir/hadoop-2.7.2
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

Save the file, and then run the following command to take these configurations into effect:

```
$ source ~/.bashrc
```

Important: Check if the `HADOOP_HOME` is correctly configured by:

```
$ echo $HADOOP_HOME
```

You should see:

```
/home/comp9313/workdir/hadoop-2.7.2
```

2. Configure HDFS

We first open the hadoop environment file, `hadoop-env.sh`, using:

```
$ gedit $HADOOP_CONF_DIR/hadoop-env.sh
```

and add the following to the **end** of this file

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

Then open the HDFS core configuration file, core-site.xml, using:

```
$ gedit $HADOOP_CONF_DIR/core-site.xml
```

Note that it is in xml format, and every configuration should be put in between <configuration> and </configuration>. You need to add the following lines:

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>file:/home/comp9313/workdir/Hadoop-2.7.2/tmp</value>
</property>

<property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
</property>
```

More configuration details please refer to:

<https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/core-default.xml>

Finally open the configuration file hdfs-site.xml, using:

```
$ gedit $HADOOP_CONF_DIR/hdfs-site.xml
```

You need to add the following lines between <configuration> and </configuration>:

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>

<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/home/comp9313/workdir/Hadoop-2.7.2/tmp/dfs/name</value>
</property>

<property>
  <name>dfs.datanode.data.dir</name>
  <value> file:/home/comp9313/workdir/Hadoop-2.7.2/tmp/dfs/data</value>
</property>
```

Now you have already done the basic configuration of HDFS, and it is ready to use.

More configuration details please refer to;

<https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml>

Start HDFS

1. Work in the Hadoop home folder.

```
$ cd $HADOOP_HOME
```

Format the NameNode (the master node):

```
$ $HADOOP_HOME/bin/hdfs namenode -format
```

You should see the output like below if successful:

```
17/03/05 02:38:52 INFO common.Storage: Storage directory /home/comp9313/workdir/hadoop-2.7.2/tmp/dfs/name has been successfully formatted.
17/03/05 02:38:52 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
17/03/05 02:38:52 INFO util.ExitUtil: Exiting with status 0
17/03/05 02:38:52 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at comp9313-VirtualBox/127.0.1.1
*****/
```

Start HDFS in the virtual machine using the following command:

```
$ $HADOOP_HOME/sbin/start-dfs.sh
```

If you see below,

```
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is a9:28:e0:4e:89:40:a4:cd:75:8f:0b:8b:57:79:67:86.
Are you sure you want to continue connecting (yes/no)? yes
```

you just need to input “yes” to continue.

2. Use the command “jps” to see whether Hadoop has been started successfully. You should see something like below:

```
comp9313@comp9313-VirtualBox:~/workdir/hadoop-2.7.2$ jps
4081 NameNode
4524 Jps
4231 DataNode
4415 SecondaryNameNode
```

Note that you should have “NameNode”, “DataNode” and “SecondaryNameNode”.

3. You can browse the web interface for the information of NameNode and DataNode at: <http://localhost:50070>. You will see:

Overview 'localhost:9000' (active)

Started:	Sun Mar 05 02:45:46 AEDT 2017
Version:	2.7.2, rb165c4fe8a74265c792ce23f546c64604acf0e41
Compiled:	2016-01-26T00:08Z by jenkins from (detached from b165c4f)
Cluster ID:	CID-1810e851-dd18-4d52-8890-d2ce72c84959
Block Pool ID:	BP-1414221996-127.0.1.1-1488641932296

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 30.56 MB of 59.88 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 31.78 MB of 32.88 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Using HDFS

1. Make the HDFS directories required to execute MapReduce jobs:

```
$ $HADOOP_HOME/bin/hdfs dfs -mkdir /user
$ $HADOOP_HOME/bin/hdfs dfs -mkdir /user/comp9313
```

Folders are created upon HDFS, rather than local file systems. After creating these folders, the `/user/comp9313` is now the default working folder in HDFS. That is, you can create/get/copy/list (and more operations) files/folders without typing `/user/comp9313` every time. For example, we can use

```
$ $HADOOP_HOME/bin/hdfs dfs -ls
```

instead of

```
$ $HADOOP_HOME/bin/hdfs dfs -ls /user/comp9313
```

to list files in `/user/comp9313`.

2. Make a directory input to store files:

```
$ $HADOOP_HOME/bin/hdfs dfs -mkdir input
```

Remember `/user/comp9313` is our working folder. Thus, the directory `input` is created under `/user/comp9313`, that is: `/user/comp9313/input`. Check the `input` folder exists using

```
$ $HADOOP_HOME/bin/hdfs dfs -ls
```

3. Copy the input files into the distributed filesystem:

```
$ $HADOOP_HOME/bin/hdfs dfs -put $HADOOP_HOME/etc/hadoop/* input
```

We will copy all files in the directory of `$HADOOP_HOME/etc/hadoop` on the local file system to the directory of `/user/comp9313/input` on HDFS. After you copy all the files, you can use the following command to list the files in `input`:

```
$ $HADOOP_HOME/bin/hdfs dfs -ls input
```

4. Please find more commands of HDFS operations here:

<https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/FileSystemShell.html>

and try these commands to operate the HDFS files and/or folders. At least you should familiar with the following commands in this lab:

```
get, put, cp, mv, rm, mkdir, cat
```

Running MapReduce in the pseudo-distributed mode

Now Hadoop has been configured to the pseudo-distributed mode, where each Hadoop daemon runs in a separate Java process. This is useful for debugging.

1. Run some of the examples provided:

```
$ $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar grep input output 'dfs[a-z.]+'
```

Just like the `grep` command in Linux (Please see here <http://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/> if you are not familiar), the above command executes a Hadoop MapReduce implementation of `grep`, while will find all files starting with “dfs” in the folder `input`, and output the results to the directory “output”.

2. Examine the output files. Copy the output files from the distributed filesystem to the local filesystem and examine the results:

```
$ $HADOOP_HOME/bin/hdfs dfs -get output output
$ cat output/*
```

Or, you can examine them on HDFS directly

```
$ $HADOOP_HOME/bin/hdfs dfs -cat output/*
```

You can see the results like below:

```
comp9313@comp9313-VirtualBox:~/workdir/hadoop-2.7.2$ hdfs dfs -cat output/*
8   dfs.audit.logger
4   dfs.class
3   dfs.server.namenode.
2   dfs.replication
2   dfs.audit.log.maxfilesize
2   dfs.period
2   dfs.audit.log.maxbackupindex
1   dfsmetrics.log
1   dfsadmin
1   dfs.servers
1   dfs.file
1   dfs.datanode.data.dir
1   dfs.namenode.name.dir
```

3. The `hadoop-mapreduce-examples-2.7.2.jar` is a package of classic MapReduce implementations including `wordcount`, `grep`, `pi_estimate`, etc. You can explore by checking the available applications as:

```
$ $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar
```

Choose one that you are interested in and look into the specific usage. For example, you can check the usage of `wordcount` by running:

```
$ $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar wordcount
```

You will notice that the application would need an input file and an output file as arguments, which are the inputs and outputs respectively. Thus, you can use the following command to count the frequency of words from files in our `input` folder and write the results to our `output` folder:

```
$ $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar wordcount input output
```

Warning: Note that if output already exists, you will meet an exception. You need to either delete `output` on HDFS:

```
$ $HADOOP_HOME/bin/hdfs dfs -rm -r output
```

Or, you use another folder to store the results (e.g., `output2`). Then the results can be checked using `cat` as you did before.

Execute a job on YARN

1. Configurations

If we want to run the job in a real distributed environment, we need to borrow a hand from YARN, which manages all the computing nodes and resources of Hadoop. On a single computer, we can also run a MapReduce job on YARN in a pseudo-distributed mode by setting a few parameters and running ResourceManager daemon and NodeManager daemon in addition.

We first configure the MapReduce to use the YARN framework. Open the `mapred-site.xml` :

```
$ mv $HADOOP_CONF_DIR/mapred-site.xml.template $HADOOP_CONF_DIR/mapred-site.xml
```

```
$ gedit $HADOOP_CONF_DIR/mapred-site.xml
```

and then add the following lines (still in between `<configuration>` and `</configuration>`):

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

Then open the `yarn-site.xml` to configure yarn:

```
$ gedit $HADOOP_CONF_DIR/yarn-site.xml
```

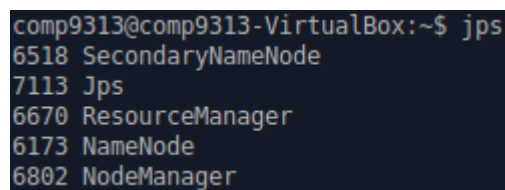
and add the following lines:

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
```

2. Start YARN:

```
$ $HADOOP_HOME/sbin/start-yarn.sh
```

3. Try `jps` again, you will see “NodeManager” and “ResourceManager”, and these are the main daemons of YARN.



```
comp9313@comp9313-VirtualBox:~$ jps
6518 SecondaryNameNode
7113 Jps
6670 ResourceManager
6173 NameNode
6802 NodeManager
```

4. Run the `grep` or `wordcount` example again.

You may observe that now the runtime is longer. Compared to the non-distributed execution, YARN is now managing resources and scheduling tasks. This causes some overheads. However, YARN allows us to deploy and run our applications in a cluster with up to thousands of machines, and process very large data in the real world.

5. Browse the web interface (for supervision and debugging) for the ResourceManager at: <http://localhost:8088/>.